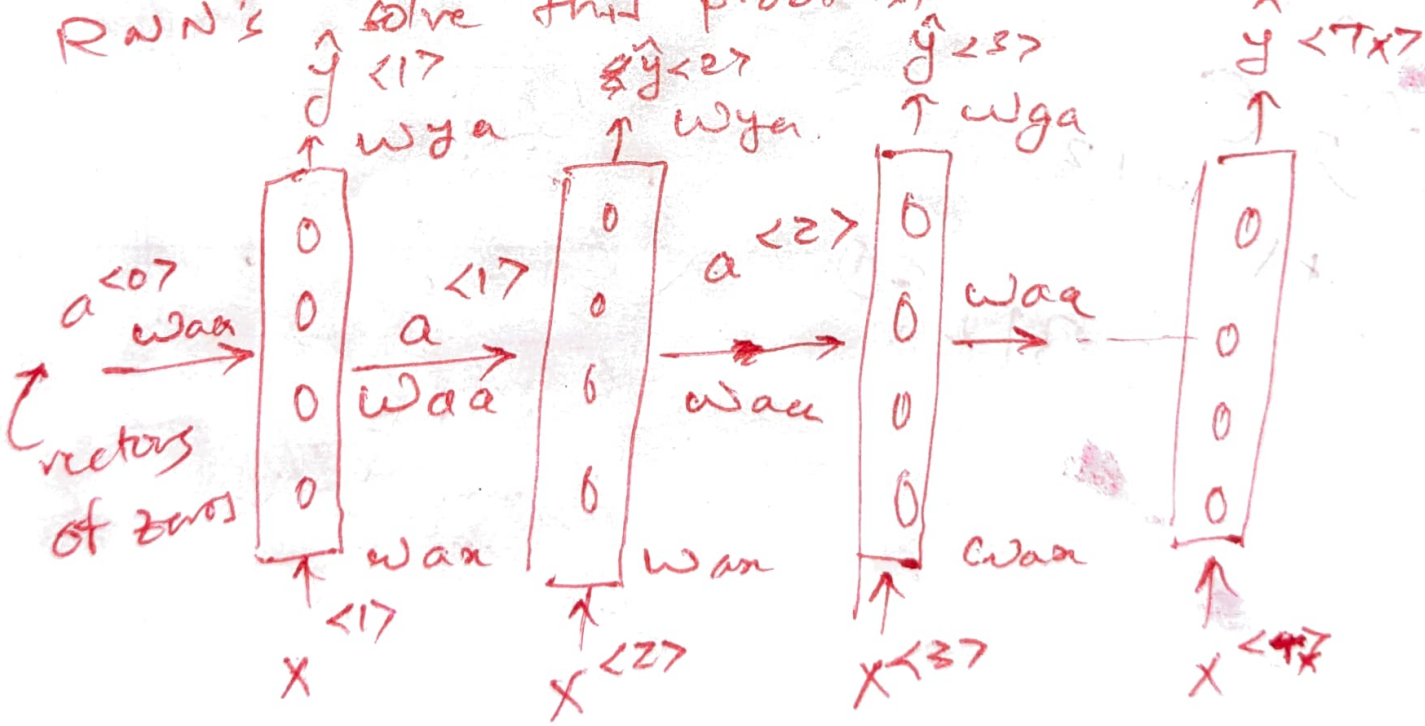
Course 5 (RNN & LSTMs)

Why we cannot use standard NN for NLP tasks?

1. Input size in standard NN is fixed, whereas, no. of words in a sentence may vary.

2. Even if we pad the ~~left~~ input which is empty (say standard input size is 100, no. of words are 70 & mark the ~~left~~ empty 30) with 0, it will require lot of computational power. (Due to one-hot encoding)

RNN's solve this problem



Wax, Way, Waa are shared across all x. i.e its same for all x.

The output of the previous input is fed ~~at~~ along with current input. i.e. prediction is only based on ~~words~~ input words in earlier sequence but not information later in the sequence. This is a drawback of this particular RNN architecture

Forward ~~prop~~ propagation

$$a^{<0>} = \vec{0}, \quad a^{<1>} = g_{(1)}(W_{aa} a^{<0>} + W_{ax} \cdot X^{<1>} + b_a)$$

$$y^{<1>} = g_{(2)}(W_{ya} a^{<1>} + b_y)$$

$g_{(1)} \rightarrow$ tanh, ReLU

? for binary classification $\rightarrow$ sigmoid

for k-way classification $\rightarrow$ softmax

General representation

① $a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax} x^{<t>} + b_a)$

② $y^{<t>} = g(W_{ay}\, a^{<t>} + b_y)$

Expression ① can be written as

$$a^{<t>} = g(W_a (a^{<t-1>}, x^{<t>}) + b_a)$$

W_a will be composed of W_{aa} & W_{ax}

say $W_{aa} = (100, 100)$ &

$W_{ax} = (100, 10{,}000)$

then $W_a = [W_{aa} \,;\, W_{ax}]$ ↕ 100

← 100 → ← 10,000 →

$= (100, 10{,}100)$

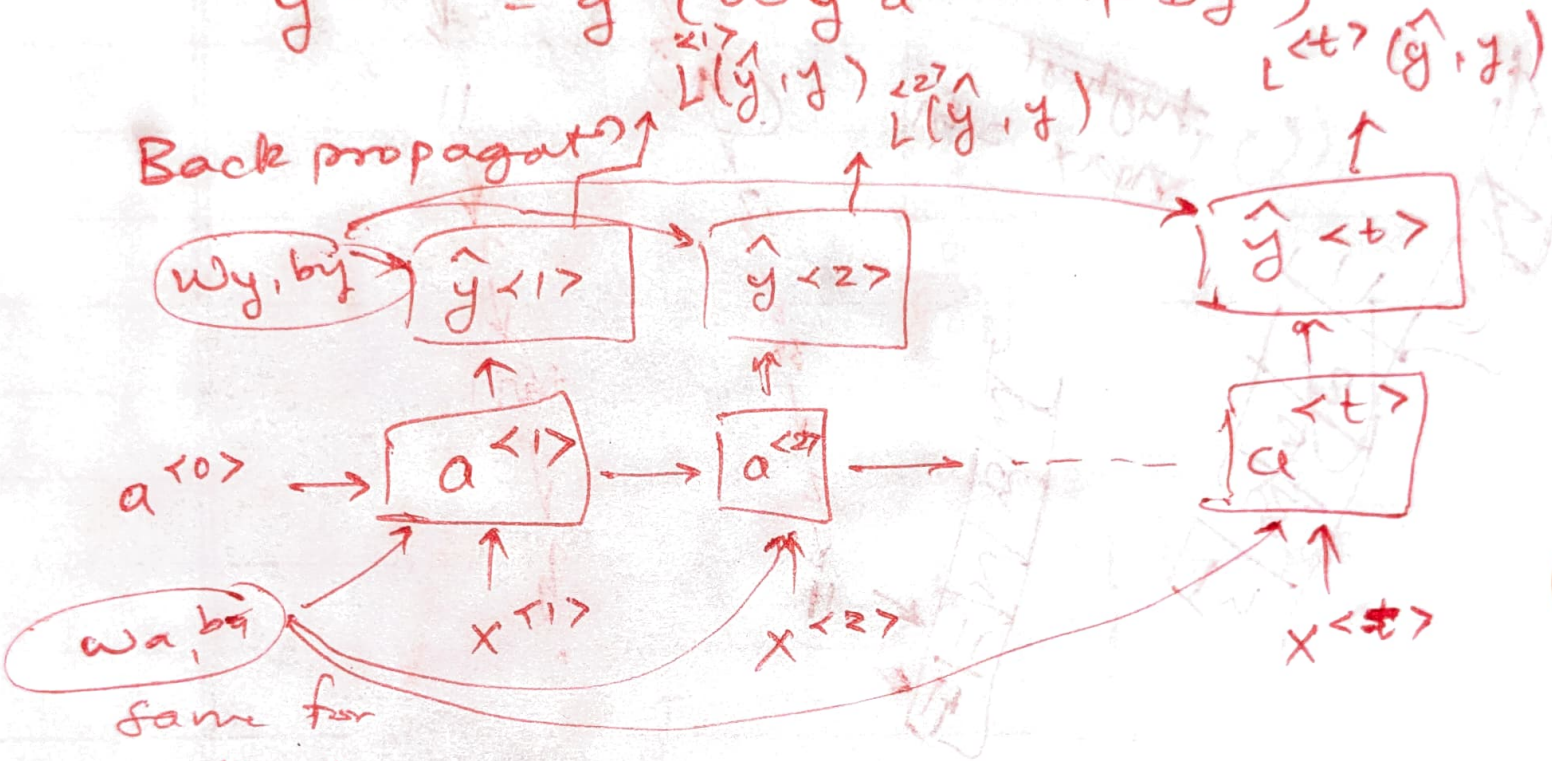
$$(a^{<t-1>}, x^{<t>}) = \begin{bmatrix} a^{<t-1>} \\ x^{<t-1>} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 10{,}000 \end{matrix}$$

$$[W_{aa} \,\vdots\, W_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>}$$

So now we can ~~use~~ use W_a as a single matrix.

i We can also write,

$$\hat{y}^{<t>} = g\left(W_y a^{<t>} + b_y\right)$$



$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \cdot \log(1 - \hat{y}^{<t>})$$

10100 for all t,

$$L(\hat{y}, y) = \sum_{t=1}^{T} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Types of RNN

- One to One (Typically, not classified as RNN)

- One to many
  Sound generatn

- many to one
  sentiment analytis (Movie review to its rating)

- Many to many
  1. $x^{<t>}$ & $y^{<t>}$ are ~~qual~~ diff.
     eg. translatn of diff. lang.
  2. $x^{<t>}$ & $y^{<t>}$ are same
     eg. name entity recognition. Check whether words in a sentence are noun or not.

Language modelling with an RNN.

Language modelling is predicting the next word / output in a sequence of words.

cats average 15 hours of sleep a ____.

predicted word

given a dictionary of words, RNN will try to predict the probability of each word.

say dictionary = {a, aaron, ---, zulu},

$\hat{y}$ ie. output will consisting of probability of each word. ~~It~~ And their probabilities sum up to 1.

The probability of each word depends upon the input & the past output. But the predicted output won't be passed as input as it may be incorrect. Hence, during <u>training</u>, the correct output is passed, instead of predicted.

$$P(y^{<t>} \mid y^{<1>}, y^{<2>}, \ldots, y^{<t-1>})$$

PAGE NO.:
DATE: / /

Sampling novel sequences

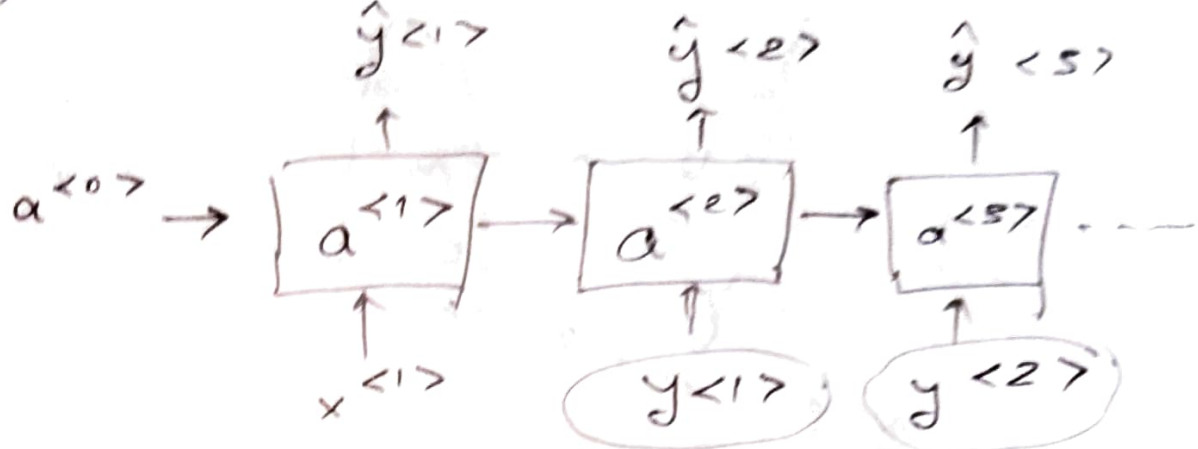
How it is different from Language modelling?

→ Sampling ~~of~~ a sequences from a trained RNN occurs in the testing phase. A trained model is used to predict the output of each word. We'll just pass ~~the~~ an empty input & we'll see, what the model generates from that input. But from a trained model, the output will be always the same. Hence, to avoid that a random word would be chosen from according to their probabilities. In sampling, the predicted o/p is passed as input to the next layer unlike modelling.

Teacher's Signature

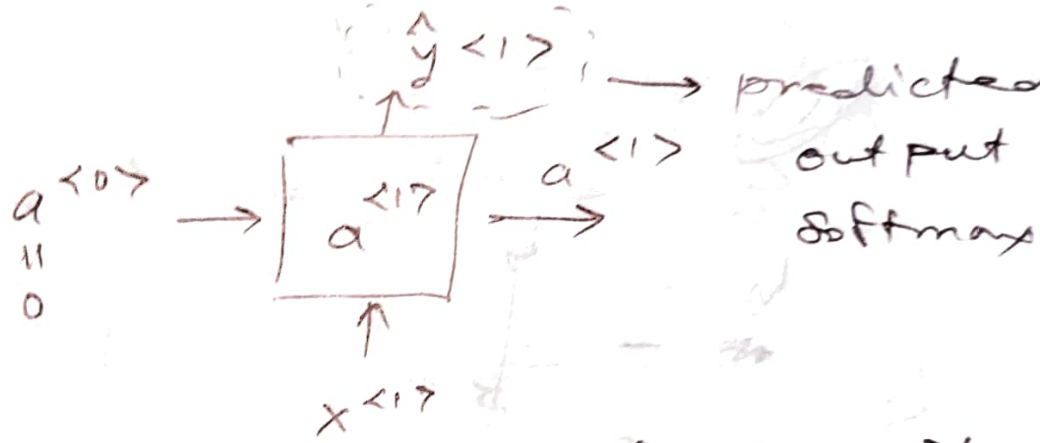
Sampling sequence from trained RNN

Training



Notice, these aren't predicted outputs. They are correct ones (from sample example).

Sampling :



→ predicted output out of softmax

$\hat{y}^{<1>}$ will be consisting of P(a), P(aaron), ... P(zulu), P(<UNK>).

Not always the character / word of highest probability is selected.

np.random.choice is used — ①

while passing $y^{<1>}$ to the next activatn funct^n, it is one hot encoded

Meaning all the words except the one which is select by ① is ~~an~~ represented by 0 & ① is represented by 1.

When to know to ~~complete~~ finish the sentence?
When $\hat{y}^{<t>}$ will generate <EOS>, the end the sentence
Also $\hat{y}^{<t>}$ may generate <UNK> i.e. unknown words. you may discard this result or you may keep it.

2 types of models
① character level language model:
Character by character is generated by each activation function. It can predict unknown words like a person's name which is not included in the vocabulary.

(2) Word level : Words are generated from each activat^n funct^n. Unknown words may not be generated

Vanishing gradients with RNN.
& exploding

if we try to include all the past sequences in the current prediction, even though they are irrelevant, we would get a very large loss at the output. & thus, ~~the~~ during backpropagation, ~~g~~ the gradients would explode. Since, the gradients use chain rule to calculate the gradients in previous layers, the gradients are multiplied. Similarly vanishing of gradient may also occurs, as the gradients may (less than 1) be small, but the gradients might too small during backpropagation making ~~no cha~~ almost no changes in the parameters. Hence to avoid this, gradient clipping is done.

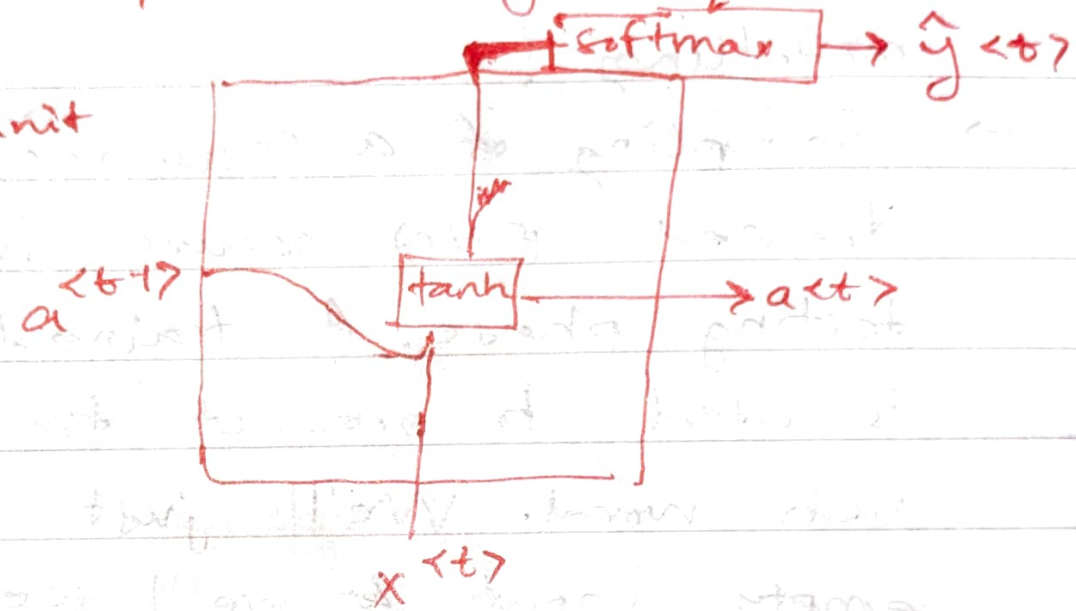
The gradients are clipped, i.e. if the ~~gra~~ past gradients are too large, they are normalized to some value, so that all the gradients are in the range of -max to max, given 'max' is the limit.

Furthermore, to avoid the unnecessary long range ~~To avoid~~ dependencies of words, we'll be using GRU & LSTM.

GRU (Gated recurrent unit)

We know, $a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$

RNN unit



c = memory cell

$c^{<t>} = a^{<t>}$ — for GRU

$\cancel{\tilde{c}^{<t>} = \tanh(W_c[c^{<t>}, x^{<t>}] + b_c)}$

$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

u → update

Γ_u will determine whether the word will be saved for further use or not. It uses a sigmoid function for that

(1) $c^{<t>} = \Gamma_u \times \tilde{c}^{<t>} + (1-\Gamma_u) \times c^{<t-1>}$

if Γ_u is 1, ~~set~~ update gate's value as new value of $c^{<t>}$.

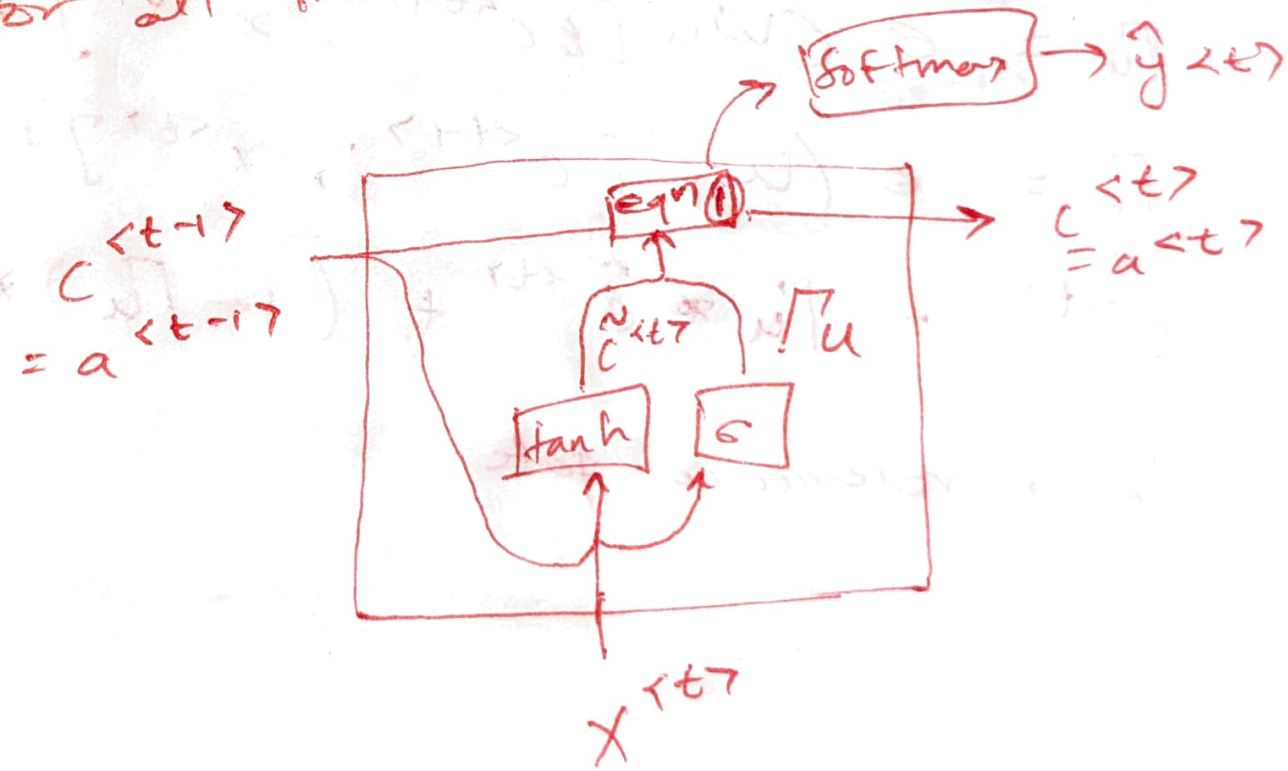
if Γ_u is 0, don't update the gate's value, keep it as it is.

eg.

The cat, which already ~~cate~~ ate ⟶ was full.

→ for cat, ~~to~~ update gate's value to 1, & for all the other words, keep it 0.

GRU



Γ_u doesn't suffer much from vanishing gradient.

$c^{<t>}$, $\tilde{c}^{<t>}$, Γ_u will have same dimensions and would depend on the dimension of activation function $a^{<t>}$

In $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$

element wise multiplication

* Full GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$$

$\Gamma_r \rightarrow$ relevance gate

LSTM

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[a^{<t-1>}, x^{<t>}\right] + b_c\right)$$

You may include Γ_r in above eqn as well, but more commonly its not part of the eqn

update $\Gamma_u = \sigma\left(W_u\left[a^{<t-1>}, x^{<t>}\right] + b_u\right)$

forget $\Gamma_f = \sigma\left(W_b\left[a^{<t-1>}, x^{<t>}\right] + b_f\right)$

output $\Gamma_o = \sigma\left(W_o\left[a^{<t-1>}, x^{<t>}\right] + b_o\right)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$\cancel{a^{<t>} = \Gamma_o * c^{<t>}} \quad a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

Bidirectional RNN

Traditional RNN & LSTM only have ~~a~~ ~~informat~~ relevance from the past & no ~~relevance~~ info. ~~of~~ from the future

① He said, "Teddy bears are on sale!"

② He said, "Teddy Roosevelt was a great President".

Here, each "Teddy" has different meaning. And we cannot decide it only from the information from the past

BRNN requires entire sequence to make prediction. Not helpful for real time application. ~~where~~ say for speech recognition, we would require to complete our sentence for BRNN to make ~~sequence~~ of words